APPARATUS AND METHOD FOR SCHEDULING PROCESSES ON A FAIR SHARE BASIS

Andrei V. Dorofeev

Andrew G. Tucker

FIELD

[0001] The present invention generally relates to techniques for managing load of a central processing unit in a computer system. The present invention also relates to techniques for allocating central processing unit percentage usage among several processes.

BACKGROUND

[0002] Modern high performance computer systems are capable of supporting multiple users and executing multiple processes or applications started by different users at the same time. Term scheduling refers to allocation of central processor unit (CPU) resources among multiple processes owned by different users in a computer system. It will be appreciated by persons of skill in the art that the fraction of the total CPU resources allocated to a particular process may depend on the number of other processes in the system and a relative importance of the process in comparison with the other processes in the system.

Another way of allocating system resources is to provide a process group, rather than a single process, with a fixed share of CPU resources. In other words, one or more processes executing on a central processing unit of a computer system can be combined into process groups, and each process group is allocated a share of the system resources that are distributed among individual processes of this process group. The system resources can be distributed among processes of the process group in an equal or unequal manner. The processes can be combined into aforementioned process groups based on various criteria, including, but not limited to, user id of the user executing the process, group id of the user executing the process, or based on any other appropriate calssification. The nature of the classification is not critical to

the present invention.

Modern operating systems include concept of processor sets, which is a group of processes executing a specific separate set of processes. Only those processes that are explicitly bound to a processor set are permitted to be executed on processors that belong to that processor set. Presently, there exists no resource allocation scheme that would leverage advantages of the processor set architecture and have an ability to distribute CPU resources among multiple process groups in a pre–defined manner.

SUMMARY

[0008] To overcome the limitations described above, and to overcome other limitations that will become apparent upon reading and understanding the present specification, apparatus, methods and articles of manufacture are disclosed that allocate a percentage of system resources among process groups in a computer system having one or more processor sets.

According to the inventive method, processes are combined into process groups based on a pre–defined criteria. Each of the process groups is assigned a number of shares representing relative importance of the group within its processor set. The inventive system allocates the system resources of the processor sets to process groups according to the number of shares assigned to a particular process group and the total number of shares of all active process groups in the processor set. A process group is considered active on a processor set if there is at least one process of this process group executing on that processor set.

DESCRIPTION OF THE DRAWINGS

[0016] Various embodiments of the present invention will now be described in detail by way of example only, and not by way of limitation, with reference to the attached drawings wherein identical or similar elements are designated with like numerals.

[0017] FIG. 1 illustrates exemplary allocation of system resources according to the inventive concept.

DETAILED DESCRIPTION

[0020] Fair share scheduling is a way to assign a particular process a fixed share of CPU resources. The fair share scheduler, as described in the white paper by J. Kay and P. Lauder is using the term share to describe the relative importance of one workload versus another.

According to one of the aspects of the present invention, various processes in the system are combined into one or more process groups. These process groups users are assigned a number of shares which represent relative importance thereof. This is a way to guarantee application performance by explicitly allocating shares (or percentage) of system CPU resources among competing workloads. Note that total number of shares assigned to all process groups need not be 100. Furthermore, to obtain the percentage of the system CPU resources available to a process group at each given moment of time, a total number of shares allocated to that process group must be divided by the total number of shares possessed by all currently active process groups. A process group is considered active when it has at least one running or runnable process. Indeed, to ensure the complete, or 100% utilization of the system, only process groups which have executing processes at a particular time should be given share of the

CPU usage. Note that such active process groups are searched across the entire system. At any given time, the percentage of the CPU allocated to a particular process group depends on the number of shares owned by all other active process groups in the system, or the process groups that have at least one executing process in the system. Therefore, in a system where processes are combined into process groups based on user id, any new logged user with a given number of shares can decrease the CPU percentage of all other actively running users.

Modern operating systems, such as a Solaris Operating System distributed by Sun Microsystems, Inc. of Palo Alto, California have a concept of processor sets. Processor set concept applies to multiple processor computers and allows the binding of one or more processors into groups of processors. Processors assigned to processor sets will run only processes that have been bound to that processor set. In other words, the aforementioned processor set is essentially a virtual single– or multi–processor computer system within a physical computer, which has its own set of running processes. The concept of processor set is especially helpful, for example, when certain important process need to be provided with a separate one or more processors. For example, in a computer system providing services to http clients, a separate processor set can be allocated to running a web server, while all other processes can be executed on a second, separate processor set. In this case, the amount of CPU resources allocated to the web server will not depend on the other processes executing in the system.

However, when the aforementioned processor sets are used in conjunction with the conventional fair share scheduler, the performance of processes running on one processor set may be impacted by the work performed by processes running on another processor set, which is

an undesirable effect.

The reason why the existing fair share scheduler does not work satisfactorily with processor sets is because that total number of shares for all active process groups is calculates across the entire system, when in fact it should be calculated only within boundaries of the current processor set. If the total number of shares is kept separate for each processor set, then the CPU allocation for a given process group will only depend on other process groups who have their active processes on the same processor set. The work done on other processor sets will be unaffected. This is more intuitive behavior of such configurations than what it has been in the past.

[0022] The inventive fair share scheduler will now be described in detail. According to the inventive concept, various processes in a computer system are combined into process groups. Each of these process groups is assigned a fixed number of shares, which is the number that represents relative importance of process groups. The number of shares of a process group is used to allocate system resources among processes of that process group executing within a predetermined processor set, in the manner described in detail below. Specifically, the inventive fair share scheduler considers each processor set to be a separate virtual computer. Different processor sets do not share processes, in other words, a process must execute on a single processor set.

In one embodiment of the invention, each process group is given the same number of shares for all processor set. It should be noted that even if process group has zero shares, processes of this process group may still be executed on processor sets, which do not have any other active process groups. Percentage of the resources of the specific processor set allocated to

processes of a particular process group is calculated as a ratio of the shares of that process group on the processor set to the total number of shares of all active process groups operating in that set. The process group is considered active on a processor set, if that processor set executes at least one process of that process group.

Now, an exemplary illustration of the inventive scheduling concept will be presented, see Fig. 1. In this illustration, a computer system includes five processors: p1–p5 and three process groups g1, g2 and g3. The processors p1 and p2 constitute processor set s1. Processors p3, p4, and p5 are combined into processor set s2. Group g1 is assigned 3 shares and it is executing on the processor set s1. Process groups g2 and g3 are assigned 1 and 2 shares respectively, and they both execute on processor set s2.

The conventional fair share scheduler described above would allocate the system resources without reference to the processor sets. Specifically, it would assign 3/(3+1+2) = 3/6 = 50% of the entire system resources to group g1. Group g2 would receive 1/(3+1+2) = 1/6 = 16.7%, while group g3 would get 2/(3+1+2) = 2/6 = 33.3% of the system resources.

The inventive system, on the other hand, group g1 receives entire processor set s1, which is 40% = 100% * 2/5 of the total system resources. Groups g2 and g3 jointly executing on processor set s2 receive 1/3 * 3/5 = 1/5 = 20% and 2/3 * 3/5 = 2/5 = 40% of the system resources, respectively. It should be noted that the percentage of the system resources assigned to process group g1 is independent of the number of the process groups and the load of the processor set s2.

[0029] While the invention has been described herein with reference to preferred embodiments thereof, it will be readily apparent to persons of skill in the art that various

modifications in form of detail can be made with respect thereto without departing from the spirit and scope of the invention as defined in and by the appended claims. For example, the present invention is not limited allocating CPU shares only. Generally, processes, users, or groups of users can be allocated shares relating to various system resources, such as CPU, disk or memory usage. In addition, shares need not be assigned to groups of users. According to the inventive concept, shares can be assigned to processes, users, or groups of users, or in any other manner. It will also be appreciated by those of skill in the art, that a number of processors in a processor set may not be an integer number. In other words, the concept of processor set should be viewed, in the context of the present invention, as allocating shares of the total processor resources of a computer system among virtual computers (processor sets), that do not share processes, rather than simply combining resources of physical CPUs. Finally, each process group could be given a separate number of shares for each processor set.

Those of skill in the art will undoubtedly appreciate that the invention can be implemented on a wide variety of computer systems including, but not limited to, general purpose computers and special purpose computers such as network appliances. As well known in the art, a computer consists at least of a central processing unit, a memory unit, and an input/output interface. The aforementioned computer components can be arranged separately, or they can be combined together into a single unit. The computer memory unit may include a random access memory (RAM) and/or read only memory (ROM). The present invention can be implemented as a computer program embodied in any tangible storage medium, or loaded into the computer memory by any known means. As an alternative to implementing the present invention as a computer program, the present invention can be also embodied into an electronic

circuit. This embodiment may provide an improved performance characteristics.